

# UTILITY PATENT APPLICATION TRANSMITTAL

Attorney Docket No.

95-427

First Named Inventor or  
Application Identifier:

WHEELER et al

Title: Generic Command Interface for Multiple Executable Routines

(Only for new nonprovisional applications under 37 CFR 1.53(b))

## APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent  
application contents.

## ADDRESS TO:

Assistant Commissioner for Patents  
Box Patent Application  
Washington, DC 20231

1. ☐ Fee Transmittal Form
2. ☒ Specification, Claims & Abstract ..... [ Total Pages: 13 ]
3. ☒ Formal Drawing(s) (35 USC 113) .... [ Total Sheets: 3 ]
4. ☐ Oath or Declaration ..... [ Total Pages:    ]
  - a. ☐ Newly executed (original or copy)
  - b. ☐ Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional with Box 17 completed)
    - i. ☐ **DELETION OF INVENTOR(S)**  
Signed statement attached deleting inventor(s) named in the prior application,  
see 37 CFR 1.63(d)(2) and 1.33(b).
5. ☐ Incorporation by Reference (usable if Box 4b is checked)  
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☒ Appendix ..... [ Total Pages: 6 ]
7. ☐ Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
  - a. ☐ Computer Readable Copy
  - b. ☐ Paper Copy (identical to computer copy)
  - c. ☐ Statement verifying identity of above copies

## ACCOMPANYING APPLICATION PARTS

8. ☐ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(b) Statement (when there is an assignee) ☐ Power of Attorney
10. ☐ English Translation Document (if applicable)
11. ☐ Information Disclosure Statement (IDS)/PTO-1449 ☐ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
14. ☐ Small Entity Statement(s) ☐ Statement filed in prior application, status still proper and desired.
15. ☐ Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. ☐ Other:

## 17. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No:        /       

## 18. CORRESPONDENCE ADDRESS

Leon R. Turkevich, Esq.  
CUSTOMER NO.: 23164  
2000 M Street, N.W., 7<sup>th</sup> Floor  
Washington, D.C. 20036-3307

Name (Type)

Leon R. Turkevich

Registration No.  
(Att./Agent)

34,035

Signature



Date

June 28, 2000

06/28/00

Jc685 U.S. PTO

Jc615 U.S. PTO

09/604880

06/28/00

09604880-062800

95-427 [WGM 1897]

Patent

UNITED STATES UTILITY PATENT APPLICATION

FOR

GENERIC COMMAND INTERFACE FOR MULTIPLE EXECUTABLE ROUTINES

INVENTORS:

Jeffrey Wheeler of Glen Allen, Virginia  
Paul Mustoe of Richmond, Virginia

PREPARED BY:

Leon R. Turkevich, Esq.  
2000 M STREET, N.W., 7<sup>th</sup> Floor  
WASHINGTON, D.C. 20036-3307  
(202) 261-1059

008250" 08340950

# GENERIC COMMAND INTERFACE FOR MULTIPLE EXECUTABLE ROUTINES

## BACKGROUND OF THE INVENTION

### FIELD OF THE INVENTION

The present invention relates to command and interface control of Operating Administration and Monitoring (OAM) executable routines within software systems.

### DESCRIPTION OF THE RELATED ART

Operating Administration and Monitoring (OAM) tools are software-based resources used as administration and/or diagnostic tools for complex processor-based executable software systems, such as software-based unified messaging software systems. A subset of OAM tools includes Real Time Monitoring (RTM) programs, used to monitor and control selected states and processes within the software based system. For example, a given RTM program may generate a real-time display (i.e., "a screen") of selected parameters during execution of a prescribed process; the RTM program may also provide a diagnostic resource that enables resetting of various states or variables within the prescribed process. Other administration and diagnostic tools include external binary files that execute in response to a procedure call, and Simple Network Management Protocol (SNMP) agents or scripts configured for generating an e-mail message as an alarm in response to a detected event.

Hence, system administrators may attempt to utilize multiple tools within a software system in order to increase the available administration and diagnostic tools for improved system performance. The use of multiple RTM programs and other OAM tools, however, requires the users to remember the names and syntaxes of numerous commands for the respective RTM programs and OAM tools. Hence, an increase in the number of OAM tools would result in the system administrator needing to develop expertise in the command names and syntaxes for the respective OAM tools.

### SUMMARY OF THE INVENTION

There is a need for an arrangement that integrates multiple RTM programs and command and control functionality for a user, without the necessity of learning the respective command formats

and syntax.

There is also a need for arrangement that enables a simple command language to be utilized for control of multiple RTM programs having respective command formats.

These and other needs are attained by the present invention, where a processor based system having a parser is configured for validating a generic command received from a user relative to a command parse tree. The command parse tree includes multiple elements, each specifying at least one corresponding generic command component and a corresponding at least one command action value. The parser, upon identifying a best match among the elements, issues a prescribed command for a selected one of the management programs according to the corresponding command format based on the selected command action value. Hence, a user may control multiple management programs having respective command formats, by using a set of generic commands that are independent from the command formats, eliminating the necessity that the user needs to learn the detailed command formats and syntax.

One aspect of the present invention provides a method in a processor-based system configured for executing a plurality of management programs according to respective command formats. The method includes receiving a generic command from the user, and validating the generic command based on a command parse tree that specifies valid generic commands relative to a prescribed generic command format, the command parse tree having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the validating step including identifying one of the elements as a best match relative to the generic command. The method also includes issuing a prescribed command of a selected one of the management programs according to the corresponding command format, based on the identified one element.

Another aspect of the present invention provides a system configured for executing a plurality of management programs according to respective command formats. The system includes a parser having a command parse tree configured for validating a generic command received from a user, the command parse tree configured for specifying valid generic commands relative to a prescribed generic command format and having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the parser

identifying one of the elements as a best match relative to the generic command. The system also includes a plurality of translators configured for issuing commands for the management programs according to respective command formats, the parser outputting a prescribed command to a selected one of the translators based on the identified one element.

5 Additional advantages and novel features of the invention will be set forth in part in the description which follows and in part will become apparent to those skilled in the art upon examination of the following or may be learned by practice of the invention. The advantages of the present invention may be realized and attained by means of instrumentalities and combinations particularly pointed out in the appended claims.

#### BRIEF DESCRIPTION OF THE DRAWINGS

Reference is made to the attached drawings, wherein elements having the same reference numeral designations represent like elements throughout and wherein:

15 Figure 1 is a diagram of a system configured for executing multiple management programs according to respective command formats based on a generic command set according to an embodiment of the present invention.

Figure 2 is a diagram illustrating in detail the parser of Figure 1 according to an embodiment of the present invention.

20 Figures 3A and 3B are diagrams illustrating the validation of generic commands by the parser of Figure 1 according to an embodiment of the present invention.

#### BEST MODE FOR CARRYING OUT THE INVENTION

25 Figure 1 is a diagram of a system configured for executing a plurality of management programs according to respective command formats according to an embodiment of the present invention. The processor based system 10 includes a user input interface 12, for example a terminal interface, that enables a user to input a generic command string, described below. The processor based system 10 also includes a parser 14 configured for validating the generic command received by the user input interface 12 from the user, and translators 16 configured for issuing commands to respective management programs 18 according to respective command formats. As shown in Figure

1, the management programs 18, implemented for example by different OAM tools such as RTM programs, may be executed within the processor based system or externally as external agents accessible using a prescribed application programming interface (API). The management programs 18 may provide different administration and maintenance functions, for example initiating various  
5 real-time screens used to monitor the internal state of executable processes within the software based system 10; alternately, different tools 18 may allow the user to control the various states within the various component of the software based system 10 via external programs (e.g., programs 18c or 18d), or may be used to issue external alarms (e.g., SNMP manager scripts) for external routines such as message waiting indicator routines.

10 A disadvantage of utilizing many different tools 18 is that each tool 18 tends to have its own screen and/or command, providing difficulties for the system administrator to determine which tool is the best tool (and/or which is the best syntax) to use for a given problem.

According to the disclosed embodiment, the parser 14 and the translators 16 provide a unified administration and diagnostic tool which incorporates the functionality of all external administrative  
15 executable binary files, RTM programs, agent manipulation scripts, and various requested snapshot queries, as well as including an extensive help system. In particular, the parser 14 and the translators 16 provide a generic command syntax that integrates the functionality of the different tools 18 and that automatically selects the appropriate command for the best tool for executing a given generic command. As illustrated in Part A. of the attached appendix, the new syntax provides a generic  
20 instruction set that provides an abstraction of the tool-specific command formats and syntax, enabling a user to issue command based on the relative functions, as opposed to the specific syntax for a corresponding tool 18.

Figure 2 is a diagram illustrating in detail the parser 14 of Figure 1 according to an embodiment of the present invention. The parser 14 includes a command word translation table 20  
25 and a command parse tree 22. The command word translation table 20 is configured for storing, for each prescribed command word 26, a corresponding token value 28 that is used by the parser 14 to identify a specific command for a selected one of the translators 16. In particular, the command word translation table 20 includes all the command words 26 that are valid according to the generic syntax, illustrated for example in Part B of the attached appendix.

The parser 14 is configured for validating a received generic command by comparing each input command word to the command parse tree 22 to determine for the received generic command a tree element 24 identified as a best match. Each tree element 24 includes at least one token-command key pair 30 that specifies a token (T) 28 and a corresponding command key (CK) 32, enabling the parser 14 to identify the appropriate prescribed command based on the command key specified for the matching token. In particular, the parser 14 recursively traverses the command parse tree 22 for each command word to identify the best match for the generic command. If only a portion of the generic command is identified as valid (e.g., only the first three command words are valid), the parser 14 selects the command key 32 for the matching token 28 from the last valid tree element 24.

Figure 3 is a diagram illustrating the method of validating a received generic command and translating the received generic command into a command for a specific management program according to an embodiment of the present invention. The operations described with respect to Figures 2 and 3 can be implemented as executable code that is stored on a computer readable medium (e.g., a hard disk drive, a floppy drive, a random access memory, a read only memory, an EPROM, a compact disk, etc). The method begins in step 40, wherein the parser begins parsing the first word of the received generic command by comparing the first input command word to the command word translation table 20 for identification of a matching token 28. For example, assume that the parser 14 receives the valid command "watch tcp connections". The parser identifies the token value "8" as corresponding to the first command word "watch". The parser 14 then traverses the command parse tree 22 in step 42 to search for the matching token 28. As illustrated in Figure 2, the parser 14 locates the matching token in the first tree element 24a. If the parser 14 determines in step 44 that the first command word is valid, the parser 14 continues searching the next command word in step 46. If the first command word is invalid based on no match in the first element 24a of the command parse tree, the parser 14 returns an invalid command message to the user in step 56.

The parser 14 then parses the next word (e.g., "tcp") of the received generic command in step 46 by locating the corresponding token 28 (e.g., "6" for "tcp") in the table 20, and then traversing in step 48 the tree elements that depend from the matched tree element 24a (e.g., 24b). The parser 14 determines a match between the token 28 ("6") corresponding to the command word "tcp" in the

token-command key pair 30d in step 50, enabling the parser to continue for the next command word. As described above, the parser 14 repeats the process in step 52 for the third command word "connections" having the token "2" and identifying a match between the entire generic command and the token-command key 30 specified in the tree element 24c. The parser 14 identifies in step 54 the prescribed command for a selected one of the translators 16 based on the value of the command key 32 within the matching token-command key pair 30 (e.g., "CK=3") of the last valid command word, which maps to a translation table that specifies a specific command for a specific translator 16.

As described above, the parser 14 can identify a command key 32 even if only a portion of the command is valid. Assume for example that the parser 14 receives the invalid command "get udp connection info". In this case, the individual command words are valid from the command word translation table 20, however, the sequence is invalid. In particular, the command word "get" having a token value of "3" reaches the token-command key pair 30b, however the command word "udp" having a token value of "7" does not reach any child of the tree element 24a. Hence, the parser 14 uses the last valid command key ("6") in step 54 based on the matching token for the first valid word located in the token-command key pair 30b. The command key is mapped to a selected one of the translators 16 in an attempt to provide a command to the corresponding resource 18. If the selected resource 18 determines that the command is invalid, the selected resource 18 at that time may prompt the user for a correct command.

The disclosed arrangement enables the use of generic commands for multiple OAM tools that have respective command syntax, resulting in a single point of entry for administering and maintaining complex software based systems. The disclosed arrangement provides the user a single set of commands and syntax to learn, facilitating the use of multiple administrative and maintenance tools.

While this invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not limited to the disclosed embodiments, but, on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.



What is Claimed Is:

1. A method in a processor-based system configured for executing a plurality of management programs according to respective command formats, the method comprising:

receiving a generic command from the user;

validating the generic command based on a command parse tree that specifies valid generic commands relative to a prescribed generic command format, the command parse tree having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the validating step including identifying one of the elements as a best match relative to the generic command; and

issuing a prescribed command of a selected one of the management programs according to the corresponding command format, based on the identified one element.

2. The method of claim 1, wherein the generic command includes at least one input command word, the validating step including:

comparing each input command word to a command word translation table, configured for storing for each prescribed command word a corresponding token, for identification of a matching token; and

determining a presence of the matching token within the command parse tree for each input command word.

3. The method of claim 2, wherein the determining step includes recursively traversing the command parse tree based on an order of the input command words for identification of the matching token within the identified one element.

4. The method of claim 3, wherein the issuing step includes issuing the prescribed command based on a corresponding command key specified for the matching token within the identified one element.

5. The method of claim 4, wherein the issuing step further includes accessing a prescribed translator configured for converting the generic command according to the corresponding command format into the prescribed command based on the corresponding command key.

6. The method of claim 5, wherein the validating step including validating at least a portion of the generic command by identifying the one element having the best match relative to the portion of the generic command, the issuing step including issuing the prescribed command based on the identified one element corresponding to the portion of the generic command.

7. The method of claim 6, further comprising executing the prescribed command within the corresponding selected one management program.

8. The method of claim 1, wherein the validating step including validating at least a portion of the generic command by identifying the one element having the best match relative to the portion of the generic command, the issuing step including issuing the prescribed command based on the identified one element corresponding to the portion of the generic command.

9. The method of claim 8, further comprising executing the prescribed command within the corresponding selected one management program.

10. A system configured for executing a plurality of management programs according to respective command formats, the system comprising:

a parser having a command parse tree configured for validating a generic command received from a user, the command parse tree configured for specifying valid generic commands relative to a prescribed generic command format and having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the parser identifying one of the elements as a best match relative to the generic command; and

a plurality of translators configured for issuing commands for the management programs according to respective command formats, the parser outputting a prescribed command to a selected one of the translators based on the identified one element.

11. The system of claim 10, wherein the parser further comprises a command word translation table configured for storing for each prescribed command word a corresponding token for identification of a matching token, the parser configured for determining a presence of the matching token within the command parse tree for each input command word.

12. The system of claim 11, wherein the parser recursively traverses the command parse tree based on an order of the input command words for identification of the matching token within the identified one element.

13. The system of claim 12, wherein the parser validates at least a portion of the generic command by identifying the one element having the best match relative to the portion of the generic command.

14. A computer readable medium having stored thereon sequences of instructions for executing a plurality of management programs according to respective command formats, the sequences of instructions including instructions for performing the steps of:

receiving a generic command from the user;

validating the generic command based on a command parse tree that specifies valid generic commands relative to a prescribed generic command format, the command parse tree having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the validating step including identifying one of the elements as a best match relative to the generic command; and

issuing a prescribed command of a selected one of the management programs according to the corresponding command format, based on the identified one element.

15. The medium of claim 14, wherein the generic command includes at least one input command word, the validating step including:

comparing each input command word to a command word translation table, configured for storing for each prescribed command word a corresponding token, for identification of a matching token; and

determining a presence of the matching token within the command parse tree for each input command word.

16. The medium of claim 15, wherein the determining step includes recursively traversing the command parse tree based on an order of the input command words for identification of the matching token within the identified one element.

17. The medium of claim 16, wherein the issuing step includes issuing the prescribed command based on a corresponding command key specified for the matching token within the identified one element.

18. The medium of claim 17, wherein the issuing step further includes accessing a prescribed translator configured for converting the generic command according to the corresponding command format into the prescribed command based on the corresponding command key.

19. The medium of claim 18, wherein the validating step including validating at least a portion of the generic command by identifying the one element having the best match relative to the portion of the generic command, the issuing step including issuing the prescribed command based on the identified one element corresponding to the portion of the generic command.

20. The medium of claim 19, further comprising instructions for performing the step of executing the prescribed command within the corresponding selected one management program.

21. The medium of claim 14, wherein the validating step including validating at least a portion of the generic command by identifying the one element having the best match relative to the portion of the generic command, the issuing step including issuing the prescribed command based on the identified one element corresponding to the portion of the generic command.

22. The medium of claim 21, further comprising instructions for performing the step of executing the prescribed command within the corresponding selected one management program.

23. A system configured for executing a plurality of management programs according to respective command formats, the system comprising:

means for validating a generic command received from a user, the validating means configured for specifying valid generic commands relative to a prescribed generic command format and having elements each specifying at least one corresponding generic command component and a corresponding at least one command action value, the validating means identifying one of the elements as a best match relative to the generic command; and

a plurality of translators configured for issuing commands for the management programs according to respective command formats, the validating means outputting a prescribed command to a selected one of the translators based on the identified one element.

24. The system of claim 23, wherein the validating means comprises a command word translation table configured for storing for each prescribed command word a corresponding token for identification of a matching token, the validating means configured for determining a presence of the matching token for each input command word.

25. The system of claim 24, wherein the validating means recursively validates each input command word based on an order of the input command words for identification of the matching token within the identified one element.

95-427

## GENERIC COMMAND INTERFACE FOR MULTIPLE EXECUTABLE ROUTINES

## ABSTRACT OF THE DISCLOSURE

A processor based system having a parser is configured for validating a generic command received from a user relative to a command parse tree. The command parse tree includes multiple elements, each specifying at least one corresponding generic command component and a corresponding at least one command action value. The parser, upon identifying a best match among the elements, issues a prescribed command for a selected one of the management programs according to the corresponding command format based on the selected command action value. Hence, a user may control multiple management programs having respective command formats, by using a set of generic commands that are independent from the command formats, eliminating the necessity that the user needs to learn the detailed command formats and syntax.

The diagram illustrates a system 10. It begins with an input block 12, which feeds into a Parser block 14. The output of the Parser is distributed to a vertical stack of four Translator blocks: Translator 1, Translator 2, Translator 3, and Translator 4. Below these translators are vertical ellipsis dots. Each translator is connected to a corresponding output block: Translator 1 to 18a, Translator 2 to 18b, Translator 3 to 18c, and Translator 4 to 18d. Additionally, the stack of translators is connected to a vertical stack of API blocks (API 1, API 1, API 1, API 1), which are also connected to the output blocks 18a, 18b, 18c, and 18d respectively.

Figure 1



Word	Token
connection	1
connections	2
get	3
info	4
set	5
tcp	6
udp	7
watch	8
< Invalid Token>	-1

↑  
26

↑  
28

← 20

Root

← 22

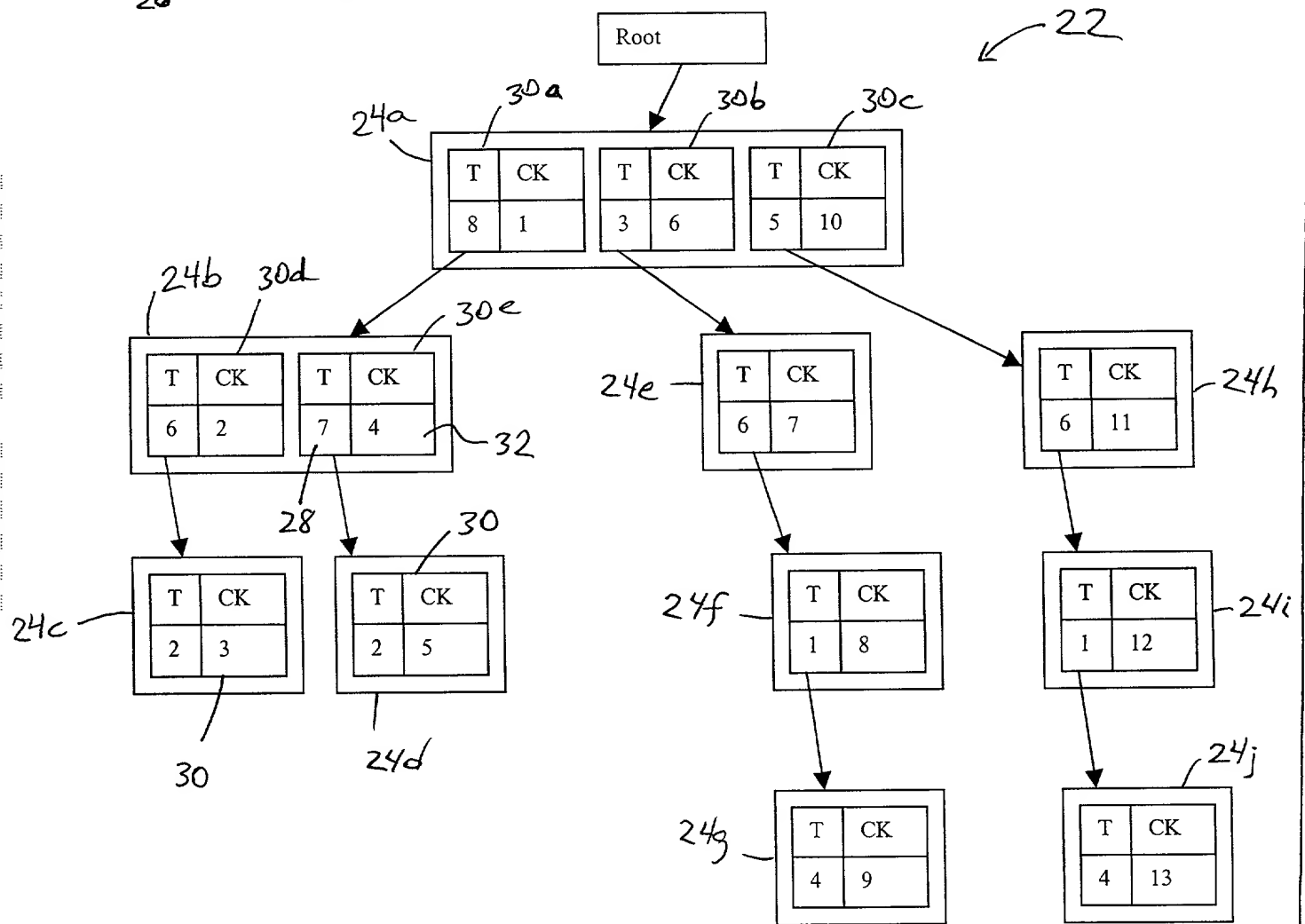


Figure 2

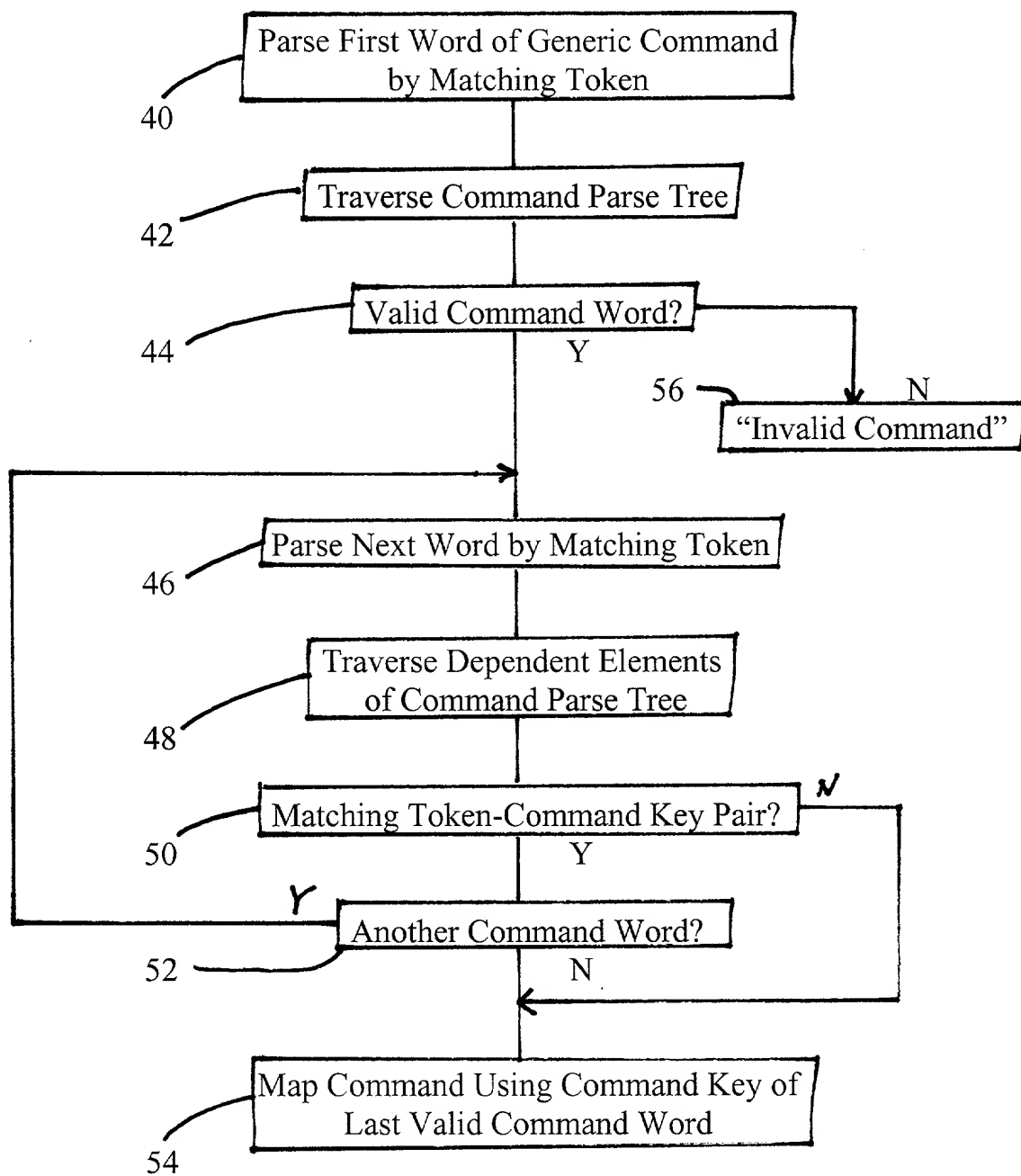


Figure 3

[illegible][illegible]

<i>Functional Item</i>	<i>New Syntax</i>	<i>Old Command Line/Syntax</i>
Get current logging level for APP	get loglevels	- None - ?
Set error logging level for APP	set loglevel acb error <off   local>	loglevel APP error <off   local>
Set warning logging level for APP	set loglevel acb warning <off   local>	loglevel APP warning <off   local>
Set info logging level for APP	set loglevel acb info <off   local>	loglevel APP info <off   local>
Set debug logging level for APP	set loglevel acb debug <off   local>	Loglevel APP debug <off   local>
Get current logging level for TNT	get loglevels	- None - ?
Set error logging level for TNT	set loglevel cma error <off   local>	loglevel TNT error <off   local>
Set warning logging level for TNT	set loglevel cma warning <off   local>	loglevel TNT warning <off   local>
Set debug logging level for TNT	Set loglevel cma debug <off   local>	loglevel TNT debug <off   local>
Get Help for RTM functions	help watch	<Object>view -?
Get Help for SNMP functions	help [start   stop   quiesce ]	obs
Get Help for reload functions	help reload	reloadsched (no parms, no help) reloadstatic
Get Help for logging functions	help set	loglevel
Get Help for query functions	help get	Obs
Get Help on help	help help	
Exit utility	exit	Q key on RTMs

## APPENDIX PART B: Generic Command Examples

### Watch

This command displays the requested RTM screen

Command Usage: Watch <Object> [<Screen>]

Valid Object/Screen Pairs:

<i>Obj</i>	<i>Valid Screens</i>	<i>Description</i>
acb	globals	Displays ACB Global counters information
	entries	Displays ACB Entry information
	states	Displays ACB States information
	comm	Displays ACB Communication Layer information
	threads	Displays ACB Thread information
cma	groups	Display group information
	sessions	Display session information
	scr	Display CMA System Call Router information
h323	entries	Display the full H323 information screen
	radvision	Display the full radvision information screen
	states	Displays the combined (H323/Rad) states info (Note: CMA allocates double the MaxPorts configurable in H323.ini, so this screen will show double the MaxPorts entries)
sms		Displays SMS information
faxprint		Displays FaxPrint process information
notify		Displays MWI_OnOff Notification info

### Get

This command allows the user to request certain system variable values.

Command Usage: get <Variable>

Valid Variables:

<i>Variable</i>	<i>Description</i>
watchtime	Gets the number of milliseconds between RTM (watch) screen refreshes
system status	Gets the current status of the system (up, down, quiesce)
loglevels	Gets the current run-time logging levels for ACB, CMA, and all loaded STATIC/STATIC_NOWAIT agents

## Set

This command allows the user to set either UMCLI variables or overall system variables.

Command Usage: set <variable> <value>

Valid Variable/Value Pairs:

<i>Variable</i>	<i>Valid Values</i>	<i>Description</i>
watchtime	Numeric in milliseconds	Sets the refresh time for RTM screens in milliseconds. Any values less than 500 will be set to 500.
loglevel acb error	"off" or "local"	Turns ACT Error level logging off or local
loglevel acb warning	"off" or "local"	Turns ACT Warning level logging off or local
loglevel acb info	"off" or "local"	Turns ACT Info level logging off or local
loglevel acb debug	"off" or "local"	Turns ACT Debug level logging off or local
loglevel cma error	"off" or "local"	Turns CMA Error level logging off or local
loglevel cma warning	"off" or "local"	Turns CMA Warning level logging off or local
loglevel cma info	"off" or "local"	Turns CMA Info level logging off or local
loglevel cma debug	"off" or "local"	Turns CMA Debug level logging off or local
loglevel <PID> error	"off" or "local"	Turns <PID> Error level logging off or local (where <PID> is the PID of any scheduled agent)
loglevel <PID> warning	"off" or "local"	Turns <PID> Warning level logging off or local (where <PID> is the PID of any scheduled agent)
loglevel <PID> info	"off" or "local"	Turns <PID> Info level logging off or local (where <PID> is the PID of any scheduled agent)
loglevel <PID> debug	"off" or "local"	Turns <PID> Debug level logging off or local (where <PID> is the PID of any scheduled agent)

## Start

This command allows the user to start various system agents or the entire system.

Command Usage: start <Agent>

Valid Agents:

<i>Agent</i>	<i>Description</i>
system	Starts the entire system
acb	Starts a manually stopped or quiesced ACB agent
cma	Starts a manually stopped or quiesced CMA agent (new functionality coming to allow quiesce of CMA ... start will be implemented with a spobjstate CMA R command)
logging	Starts the logging subsystem

[illegible]

Command Usage: stop <Agent> <Screen>

<i>Agent</i>	<i>Description</i>
system	Stops the entire system (including Logging)
acb	Stops a running or quiesced ACB agent
cma	Stops a running or quiesced CMA agent
logging	Stops the logging subsystem only if the ACB agent is down

This comand allows the user to quiesce the APP or TNT agent.

Valid Agents:

<i>Agent</i>	<i>Description</i>
acb	Quiesces acb
cma	Quiesces a running CMA agent

This command allows the user to reload various configuration files.

Command Usage: reload <Agent> [<Parameters>]

<b>Application</b>	<b>Parameters</b>	<b>Description</b>
cmaloglevels		Causes CMA to reload the DBG and TRACE sections of the \$PARMLIB/TNT/parms/TNT.ini configuration file.
Dialmap		Causes CMA to reload \$PARMLIB/TNT/parms/DialMap.ini
Route	“acb” or “cma”	Causes ACB to reload either \$PARMLIB/parms/APP/Route.<hostname> (acb) or \$PARMLIB/parms/TNT/Route.<hostname> (cma)
Sched	“all” or “static <token>”	Causes ACB to reschedule either all or a single agent defined in \$PARMLIB/parms/APP/Schedule.<hostname>. (Use of the static <token> parameter requires that <token> had previously been scheduled either as a STATIC or STATIC_NOWAIT.)

## Help

This command allows the user to get help on the valid commands available, their usage, and what they mean.

Command Usage: help [<command>]

Valid Commands:

<i>Commands</i>	<i>Description</i>
<None>	Help without any parameters gives the users either a list of the top level commands (help set to "short") or the top level commands and all valid sub commands under each top level commands (help set to "full")
watch	Gives the user a list of valid screens and what each one is
get	Gives the user a list of valid variables to query
set	Gives the user a list of valid variables and valid values for each variable
start	Gives the user a list of valid Agents to start
stop	Gives the user a list of valid Agents to stop
quiesce	Gives the user a list of valid Agents to quiesce
reload	Gives the user a list of valid Configurables (and possibly optional parameters) to reload.
help	The ubiquitous help on help (probably not necessary...)